Milad Mikeal

11/14/18

CS 162

Lab 10

Expectations:

I have already read that iterative versions of a function are a lot more efficient than recursive versions of a function. Therefore, I clearly expect the iterative version of Fibonacci to have less time clicks. I'm not sure exactly how much more efficient, but I will say that I expect the efficiency gap to be much larger for larger numbers.

N	**Recursive Time**	**Iterative Time**
5	0.044	0.014
10	0.485	0.031
20	50.075	0.081
30	5095.56	0.101
40	607470	0.149
50	n/a	0.134
100	n/a	0.301

Analysis:

As expected, the iterative version was significantly more efficient as the numbers got larger. The time clicks of the iterative version of Fibonacci remained significantly low. In fact, it only for above 20 when N was 100. Now I realize that despite the fact that recursion looks cool and nice, iterative functions are just more practical. I think it's important to learn about recursion and understand the logic behind it, but realistically, it's not too practical.

When I got to 50 for N, I realized that long type was not enough memory for the result. I already switched it from int to long. I was really surprised to realize long wouldn't have enough memory. So to avoid any problems, I changed it to unsigned long long. The iterative version of Fibonacci remained incredibly efficient. The recursive version, not so much. I had doubts about attempting to calculate Fibonacci for N=50 and N=100 recursively, and to be honest I was right. Quite frankly, I got really impatient and just cancelled the process. Put it this way, I sat in my seat for several minutes waiting for the recursive output for N=50 and I still didn't get an output. In conclusion, recursion looks nice, but speed matters.